```cpp
// posibile solutii pentru subiectele de concurs
// la examen au fost acceptate si alte solutii corecte
// solutiile au fost implementate la nivelul cunostintelor de liceu
// implementarile pot fi imbunatatite...

//Subiectul I, problema 1: Triunghiul lui Pascal

#include <iostream>
using namespace std;

	//determina elementele din linia r (din triunghiul lui Pascal) si le memoreaza in
	//sirul linie
void triunghiPascal(int r, int linie[]){
	//se calculeaza, pe rand, cate o linie din triunghi pe baza liniei anterioare
	//si se schimba liniile
	int linieNoua[33];
	linie[0] = 1; linie[1] = 1; //a 2-a linie din triunghi
	for (int ri = 2; ri <= r; ri++){
		linieNoua[0] = 1;
		linieNoua[ri] = 1;
			//se calculeaza noua linie pe baza liniei superioare ei (ca indice in triunghi)
		for (int i = 1; i < ri; i++){
			linieNoua[i] = linie[i - 1] + linie[i];
		}	//for i
			//se copiaza linia noua in locul celei vechi
		for (int i = 0; i <= ri; i++){
			linie[i] = linieNoua[i];
		}//for i
	}//for ri
}

//------------------------------------------------------------------------------
// Functia de afisare si programul principal e doar pentru test si
// nu au fost luate in calcul la notare

	//afiseaza pe ecran elementele unui sir s cu l numere intregi
void afisare(int l, int s[]){
	for (int i = 0; i < l; i++)
		cout << s[i] << " ";
	cout << endl;
}

	// Programul principal
int main(){
	int val;
	int rand[33];
	cout << "Introduceti nr linei (din triunghi): ";
	cin >> val;
	triunghiPascal(val, rand);
	afisare(val + 1, rand);
	return 0;
}
```

```cpp
//Subiectul I, problema 2: Virusi

#include <iostream>
using namespace std;

		//determina nr de ore necesar distrugerii unei populatii cu n virusi,
		//pentru un nr critic de supravietuire k
int virusi(int n, int k){
	bool distrus = (n < k);
	int nrOre = 0;
	while (!distrus){
			//daca avem nr par de virsui, injumatatim populatia
		if (n % 2 == 0)
			n = n / 2;
			//daca avem nr impar de virsui, marim populatia cu un virus
		else
			n = n + 1;
		nrOre = nrOre + 1;
			//verificam daca populatia dispare
		distrus = (n < k);
	}
	return nrOre;
}


//------------------------------------------------------------------------------
// Programul principal e doar pentru test si nu a fost luat in calcul la notare

int main(){
	int nrVirusi;
	int nrCritic;
	cout << "Nr virusi = "; cin >> nrVirusi;
	cout << "Nr critic = "; cin >> nrCritic;
	cout << "Sunt necesare " << virusi(nrVirusi, nrCritic) << " ore" << endl;
	return 0;
}
```

```cpp
//Subiectul I, problema 3: produs maxim a trei numere
#include <iostream>
using namespace std;

		//ordoneaza crescator (prin insertie) un sir s cu n numere intregi
void ordonareInsertie(int n, int s[]){
	for (int i = 1; i < n; i++){
		int j = i - 1;
		int aux = s[i];
			//se cauta locul unde trebuie inserat s[i] a.i. sirul s sa ramana ordonat
		while ((j >= 0) && (aux < s[j])){
			s[j + 1] = s[j];
			j--;
		}	//while
		s[j + 1] = aux;
	}//for i
}

	//identifica 3 elemente (a, b si c) in sirul x cu n numere intregi al caror produs este maxim
void produsMaxim(int n, int x[], int &a, int &b, int &c){

	//in sirul minime se vor retine (ordonate crescator) cele mai mici 3 elemente din sirul x
	int minime[3];
	minime[0] = x[0];
	minime[1] = x[1];
	minime[2] = x[2];
	ordonareInsertie(3, minime);

	//in sirul maxime se vor retine (ordonate crescator) cele mai mari 3 elemente din sirul x
	int maxime[3];
	maxime[0] = x[0];
	maxime[1] = x[1];
	maxime[2] = x[2];
	ordonareInsertie(3, maxime);

	for (int i = 3; i < n; i++) {
		if (x[i] < minime[2]){
			minime[2] = x[i];
			ordonareInsertie(3, minime);
		}
		if (x[i] > maxime[0]) {
			maxime[0] = x[i];
			ordonareInsertie(3, maxime);
		}
	}//for i

		//se determina, in fc de semnul elementelor din minime si maxime, valorile a, b si c
	if ((maxime[2] > 0) && (minime[0] * minime[1] > maxime[1] * maxime[0])){
		a = maxime[2];
		b = minime[0];
		c = minime[1];
	}
	else{
		a = maxime[2];
		b = maxime[1];
		c = maxime[0];
	}//if
}
```

```
//-----------------------------------------------------------------------------------------------
// Programul principal, citirea sirului si afisarea celor 3 valori e doar pentru test si
//nu au fost luate in calcul la notare

	//citeste un sir x cu n numere intregi
void citire(int &n, int x[]){
	cout << "nr elem = ";
	cin >> n;
	for (int i = 0; i < n; i++){
		cout << "sir[" << i << "] = ";
		cin >> x[i];
	}	//for i
}

	// Programul principal
int main(){
	int nrElem = 0;
	int elemente[10000];
	int a = 0;
	int b = 0;
	int c = 0;
	citire(nrElem, elemente);
	produsMaxim(nrElem, elemente, a, b, c);
	cout << "Cele 3 elemente sunt: " << a << " " << b << " " << c << endl;
	return 0;
}
```

```
//Subiectul II Cifre impare
```

a. Dându-se un număr natural ***a***, să se determine numărul format cu cifrele impare ale lui ***a*** (în ordinea apariției lor în ***a***, adică de la stanga la dreapta). În cazul în care numărul ***a*** nu conține cifre impare, rezultatul va fi 0.

b. F(2103) returneaza 13.

```
#include <iostream>
using namespace std;

	//construieste (recursiv) un nr nou cu cifrele impare ale numarului a
	//in ordinea apritiei lor (de la stanga la dreapta)
	//daca nr a nu are cifre impare, se returneaza 0
int cifreImpareRec(int a){
	if (a < 1)
		return a;
	else{
		//calculeaza ultima cifra
		int c = a % 10;
		//daca ultima cifra este impara, o foloseste in constructia noului numar
		//si o elimina din numarul a, altfel o ignora
		if (c % 2 == 1)
			return c + 10 * cifreImpareRec(a / 10);
		else
			return cifreImpareRec(a / 10);
	}
}

//-----------------------------------------------------------------------------------------
// Programul principal e doar pentru test si nu a fost luat in calcul la notare

int main(){
	int nr = 0;
	cout << "nr = ";
	cin >> nr;
	cout << cifreImpareRec(nr) << endl;
	return 0;
}
```

```cpp
//SUBIECTUL III: palindrom ciclic

#include <iostream>
using namespace std;

	//citeste un sir a cu n numere naturale
void citireSir(int &n, int a[]){
	cout << "Dati lungimea sirului: ";
	cin >> n;
	for (int i = 0; i < n; i++){
		//presupunem ca se citesc doar numere pozitive
		cout << "a[" << i << "]=";
		cin >> a[i];
	}
}

	//verifica daca o secventa (a sirului a) care incepe pe pozitia start si
	//se termina pe pozitia end este palindrom sau nu
bool estePalindrom(int a[], int start, int end){
	int i = start;
	int j = end;
		//se verifica perechi de elemente (cu indici simetrici fata de mijlocul secventei)
		//cand se gaseste o pereche de elemente diferite se opreste cautarea
	while ((a[i] == a[j]) && (i < j)){
		i++;
		j--;
	}
		//daca s-au verificat toate perechile si toate au avut elemente egale
	if (i >= j)
		return true;
	else
		return false;
}

	//determina numarul de permutari ciclice care transforma un sir a cu n numere în palindrom
	//daca sirul nu se poate transforma in palindrom prin permutari circulare, se returneaza -1
int numarPermutari(int n, int a[]){
		//se "dubleaza" sirul, copiind toate elementele inca o data in sir
	for (int i = 0; i < n; i++)
		a[n + i] = a[i];
	bool gasit = false;
	int i = 0;
	while ((i < n) && !gasit){
			//se verifica daca secventa care incepe pe pozitia i si are n elemente
			//este palindrom sau nu
		gasit = estePalindrom(a, i, i + n - 1);
		i++;
	}
	if (gasit)
		return i - 1;
	else
		return -1;
}
```

```
	//in cazul unui numar (primit ca parametru) pozitiv, afiseaza mesajul "da" si
	//valoarea numarului
	//in cazul unui numar (primit ca parametru) negativ, afiseaza mesajul "nu"
void afisare(int nr){
	if (nr >= 0)
		cout << " Da. " << nr << endl;
	else
		cout << "Nu. " << endl;
}


	//Program principal
int main(){
	int nrElemente = 0;
	int sir[1000];
	citireSir(nrElemente, sir);
	int nrPerm = numarPermutari(nrElemente, sir);
	afisare(nrPerm);
	return 0;
}
```